STPC CLIENT

GRAPHICS MODE USER'S GUIDE

Revision 0.9

STMicroelectronics
Technoparc du Pays de Gex -B.P. 112
165, rue Edouard Branly
01637 Saint Genis Pouilly (France)

Revised January 27, 1999

STMicroelectronics values your feedback. Please send it and any recommendations you may have regarding this document to:

STPC.support@st.com **with the subject line "GWG09".**

Author: M. SANCHES

Table of Contents

TECHNICAL SUPPORT

STMicroelectronics is on the Internet with a World Wide Web (WWW) site on which product presentation, technical literature as well as product support information can be found.

A dedicated STPC section is available providing up to date hardware documentation and software tools.

The Web and e-mail addresses are:

WWW: ***http://www.st.com/stpc***

e-mail: ***STPC.support@st.com***

This page is left intentionally blank

ii Notation

CRxx means CRTC register index xx
SRxx means Sequencer register index xx
GRxx means Graphic controler register index xx
ARxx means Attribute controler register index xx
BPP means Bit Per Pixel

ii Glossary

CRTC: Cathode Ray Tube Controller, the main part of a VGA controller it controls timings generation.

S-Video/YC: Analog video format also known as YC Luminance and Chrominance.

RGB: Analog video format with Red, Gren.Blue line plus synchro, available in Europe on SCART connector (PERITEL).

iii What's in the package?

The following documents are provided with this guide:

- **Pcxxx_09.pdf** VGA controller part of the STPC databook

- **STv0118.pdf** STv0118 Digital Encoder Databook (from STMicroelectronics)

- **Genmode** utility/source (for BorlandC/C++3.1) which is used to generate a '.h' C include file use in the Setm.c bios less graphic mode init code

- **Setm** utility/source to setup your graphic mode based on a include file generated using Genmode tool

- **Inis** screen init utility/source which includes miscellaneous source code to read/write vga register collect info... etc.

- **Vgainfo** utility to easily collect and manipulate vga parameters split in several registers

- **Over** utility and source to setup the overscan color

- **Cr**, **Sr**, **Gr**, **Ar** utilities to access CRTC, sequencer, graphic, attribute controller (and look up table not available) registers

- **Dclk** (v1.3) tool to program and read dot_clock frequency

- **Stv** utility and source to program the STv119

- **Ff** utility to setup the flicker filter

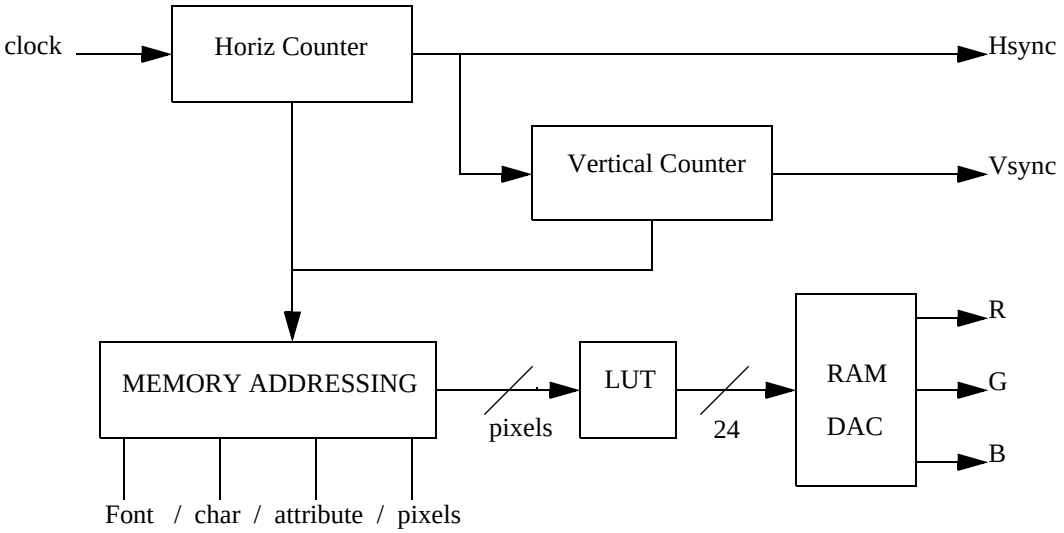
- **Vmode** utility to initialize a graphic mode with bios.

1. Introduction

Whatever below is supposed to make life easier to build your custom graphic mode, you will see it is even easier to modify an existing mode. But what on this document may not apply to standard VGA mode with strange bits plane addressing mode nor directly to text mode, it only applies to non VGA 8,16,24 BPP mode.

Also this document does not intend to teach you how to set up the vga from scratch to produce the mode you want. It is much more a guideline and some basic knowledge that should enable you to customize existing working VGA mode with the provided tools and utility program.

2. VGA Overview



3. Clock Generation

The clock path is not really complicated, the 2 main dot clock sources are internal PLL and external signal (*1).
The internal PLL provides support for 4 different independent clocks from 8MHz up to 135MHz. The 4 clock synthesizers are programmable using SIP registers with index 0x42 to 0x49. Index 0x42 & 0x43 control generator 0, 0x44 & 0x45 generator 1,... and so on. Each registered pair holds the M,N,P value to be used in the formula below to compute the frequency:

$$F\_out = \frac{14.318Mhz \bullet N}{M \bullet 2^{P}}$$

3.1. Synthesizer Register Definition

DCLK control register 00 Index 42

Bit 7 This is unused.
Bits 6-3 This the 4-bit M (divisor) value of the Dot clock synthesizer.
Bits 2-0 These are bits 7-5 of the 8 bit N (multiplier) of the Dot clock synthesizer.

DCLK control register 01 Index 43

Bit 7 This is the bit 0 of the 3-bit P (exponent) value of the Dot clock synthesizer.
Bits 6-2 These are bits 4-0 of the 8-bit N (multiplier) value of the Dot clock synthesizer.
Bits 1-0 These are bits 2-1 of the 3-bit P (exponent) value of the Dot clock synthesizer.

The other registers 0x44 to 0x49 have the same definition.

3.2. Clock selection

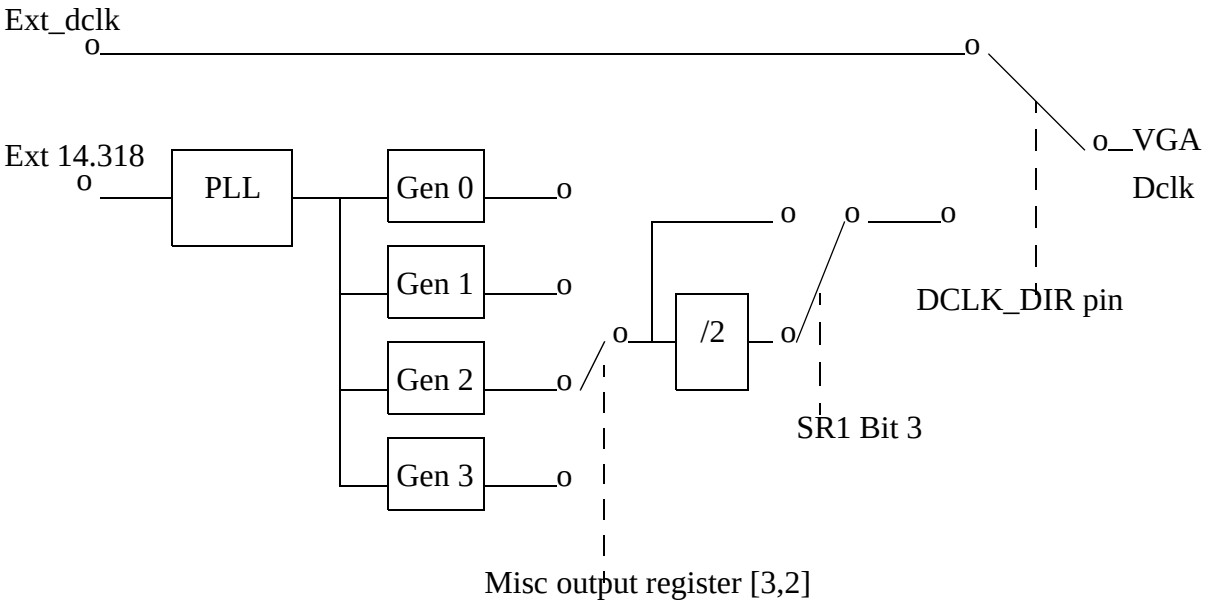
The selection between external and internal PLL dot_clock is not software controllable. It only depends on the DCLK_DIR pin level. It might nevertheless be possible to swap from internal to external clock by soft if there is any GPIO or other control logic connected to the dclk_dir pin on the board, provided that the STPC chip is at least an SIP103 (*1).

The Misc. output register (0x3C2 0x3CC R/W) bit [3,2] selects which of the 4 internal synthesizers drives the vga core input dot_clock, when internal PLL is in use.

3.3. Clock divided by 2

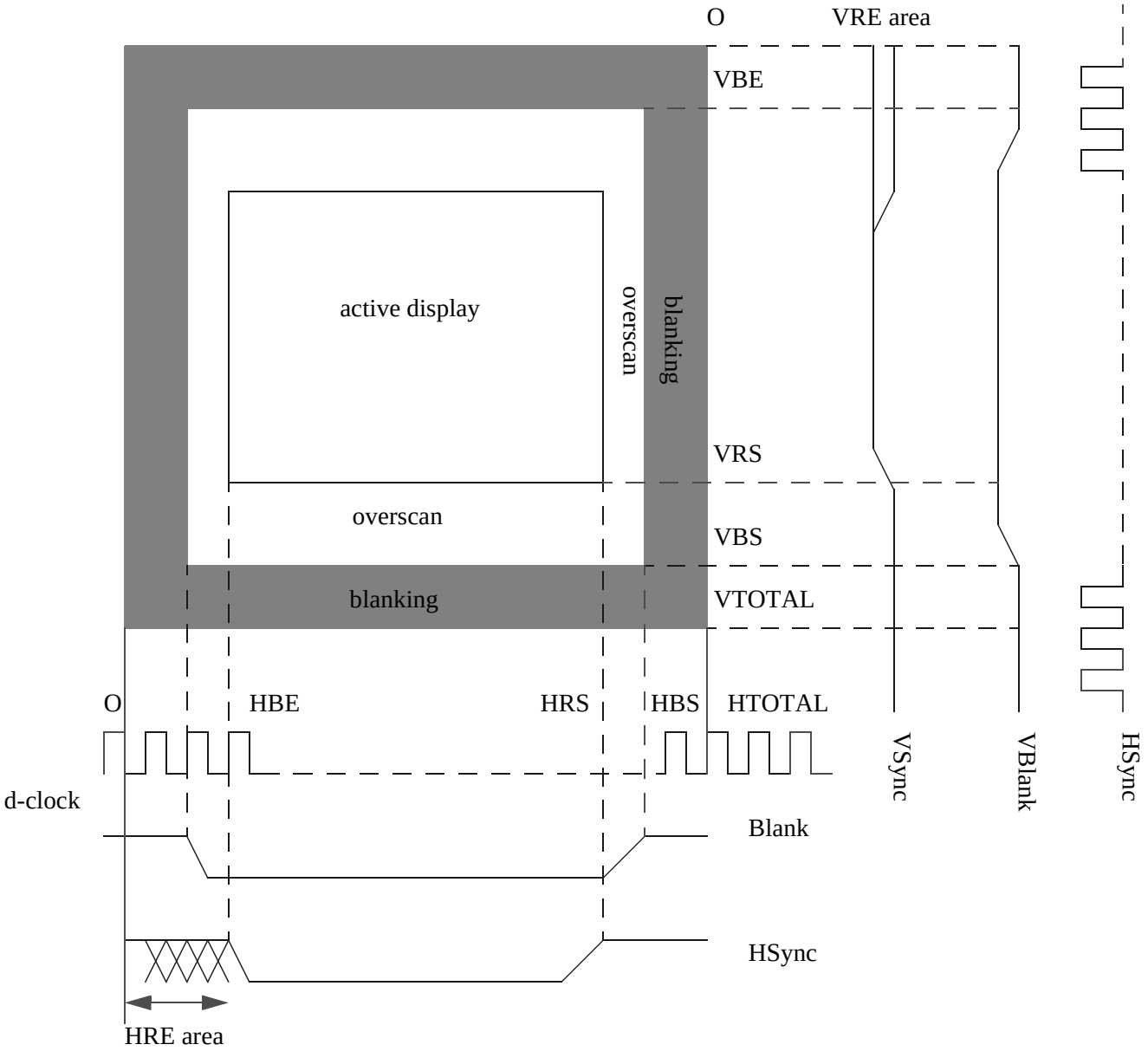
The standard Sequencer Register index 1 (SR1) bit 3 enable/disable use of a divider by 2 on the input dot-clock. These bits have no effect when using external dot clock. The critical effect is that graphic mode using this feature will not work with external dot_clock unless it is twice the value expected when using internal dot_clock. Also these bits do not allow a slower clock than 8Mhz when using On Chip PLL.

3.4.Clock data path



(*1) SIP103 or above chip revision is required to use external dot_clock. Please refer to external marking of the STPC chip.

4. Timings


O
VRE area
VBE
active display
overscan
blanking
VRS
overscan
VBS
blanking
VTOTAL
O
HBE
HRS
HBS
HTOTAL
VSync
VBlank
HSync
d-clock
Blank
HSync
HRE area

4.1. Horizontal timings

4.1.1. Definitions

All the horizontal timings are given in character unit, they must be multiplied by the character width (8 or 9) to convert them into pixel unit.

Horizontal timings are:

1) Horizontal Total, HTOT:
Define the number of character by line. Data stored in registers is HTOT -5:

2) Horizontal Display End, HDE:
This is the number of active (displayed) characters.

3) Horizontal Retrace Start, HRS:
At with characters the HSync signal start (become active).

Trick: This formula is OK most of the time but never put less than HDE.

HRS = HTOT-0x0A

4) Horizontal Retrace End, HRE:
The character at which the HSYNC signal becomes inactive.

5) Horizontal blanking starts HBS:
As with characters the output becomes blank.

Trick: typically it can be put at the same value as HDE or HRE so all non active/overscan pixels are blanked.

6) Horizontal Blanking End, HBE:
At witch character the blanking signal becomes inactive.
Trick: As a rule put 0 to make sure that blanking does not bother the active display by driving blank during active pixel, this happens if the pulse with + blank start is too big.

4.1.2. Horizontal operating clocking mode

- Count by 2 mode: (this mode is enabled by setting CR17 bit 3)

When the count by 2 mode is active the memory address counter is increased each 2 character clock which means every pixel is displayed twice. The direct effect is to divide the horizontal resolution by 2.
As an example a 320 raw mode can be displayed with the same Horizontal timing and dot clock of the same 640 raw mode.

It should be noted that when using this mode, the horizontal display end is not exactly the horizontal number of "software active pixel" but twice this value.

- Count by 4 mode: (this mode is enabled by setting CR14 bit 5)

This clocking mode is similar to Count by 2 except that the memory address counter is increased each 4 clocks instead of each 2. So it divides the Horizontal 'soft resolution' by 4.

4.2. Vertical Timings

4.2.1. Definitions

The vertical timings are given in character height unit, and must be multiplied by the correct height value to have them in pixel line unit.

1) Character Cell Height, CCH:

The character height is stored in CRTC register index 9, as number of line/character -1.
These parameters are meaningless for graphic mode (no text), which is why graphic mode use 1 for this value. It might be 2 in certain conditions (200 line mode displayed as 400 screen lines).

2) Vertical Total, VTOT

Defines the number of total lines.

3) Vertical Display End, VDE

This is the number of active (displayed) lines.

4) Vertical Retrace start, VRS

At with line the VSync signal start (becomes active).

5) Vertical Retrace End, VRE: (to be added to tools)

The line at witch the VSYNC signal becomes inactive.

6) Vertical blanking start VBS

At witch line the output become blanked.

Trick: typically it can be programmed at the same value as VDE or VRE so that all non active/overscan lines are blanked.

7) Vertical Blanking End, VBE

At which line the blanking becomes inactive.

Trick: put 0.

4.2.2. Vertical timings Mode

Double vertical Total:
When this feature is enabled all the vertical timings are doubled. The effect is to get something twice as told, to get the same thing all the vertical timings must be divided by 2. So Double Vertical Total might be useful to implement high vertical resolution mode that do not fit in standard register.

Scan Double:
When using scan double mode every line is duplicated, it might be useful to display a 200 lines mode in a 400 lines screen.

The CCH parameter can be used to implement 1 'soft active' line in more than one scan line.

5. Memory addressing

5.1. Start Address

It defines the Frame buffer offset of the 1st displayed pixel (x,y = 0,0). The start address is in most cases 0, but like for text mode it may be reprogrammed to implement several pages (by the time it does not overflow the graphic memory).

5.2. Address Wrap

CRTC extended register 0x1A hold a bit that enabled VGA addressed wrapping. When enabled the CRTC can only address 256K of memory. So whenever the memory accessed is more this bit should be set to 0.
Disabling the address wrap does not bother even for standard vga mode.
Note: The Genmode tool always disables wrapping.

5.3. Offset

It defines the byte pitch of a line. It is the amount added to the memory address to move from pixel x of the current line to address the same pixel x of line + 1. The offset is not exactly the line size in, it may be something based on that, like the size of the line but in 1 bit plane (full line size/4), or in text mode it's exactly half the size of a line seen in B000:0 (it is the size of character only not counting attribute), finally for graphic mode it must be multiplied by the character width to get the exact line size.

But even if the offset is not exactly the line size but these quantities /2 or multiply by some factor. it is not required to be always equal to these quantities. It can be greater anyway different but never less and this is for 2 main reasons:

Speed up software address computation, put a power of 2 to use shift instead of multiply:

Example 640x480x8: is 640 byte per line

so, to get memory address of pixel x,y the calculation is: 640 * y + x if 1024 is used the collation becomes: (Y << 10)+x which is at least 3 times faster.

The other reason is to match some other hardware limitation, the 2D accelerator for example cannot work with byte granular line pitch, it only has a limited set like 128,256,512,1024, ...etc.

5.4. Linear addressing

The easiest way to access the graphic memory is to use the Frame buffer technique.
The Frame buffer access is enabled by the CRTC graphic extended enable register (CRTC index 0x1F). There are, in fact, 2 frame buffers: the on screen and the off screen. The CPU address where they can be seen is defined by the Extended Graphic GBase register as follows:

On Screen FB = 128 MBytes (GBASE << 24)+ 8 MBytes
Off Screen FB = 128 MBytes (GBASE << 24);

Within the Frame buffer pixels are read and written in the same way. There is no specific VGA read/write mode with ROP operation and bit masking. Since the memory organization is linear each pixel follows one after the other and line after line, like a big array.

For 8BPP mode each pixel is a byte
For 15/16 BPP mode each pixel is a word
For 24 BPP mode each pixel is a 3 byte (R G B)

5.5. Windowed A000:0 addressing

Under some OS it may not be easy to access memory at some 128Meg so the only other way is to use windowed access trough A000:0. The addressing mode in this case is like VGA, which means some registers must be correctly set or read or written to A000 will not return not set what expected.

Here is the setting to have a linear mapping:

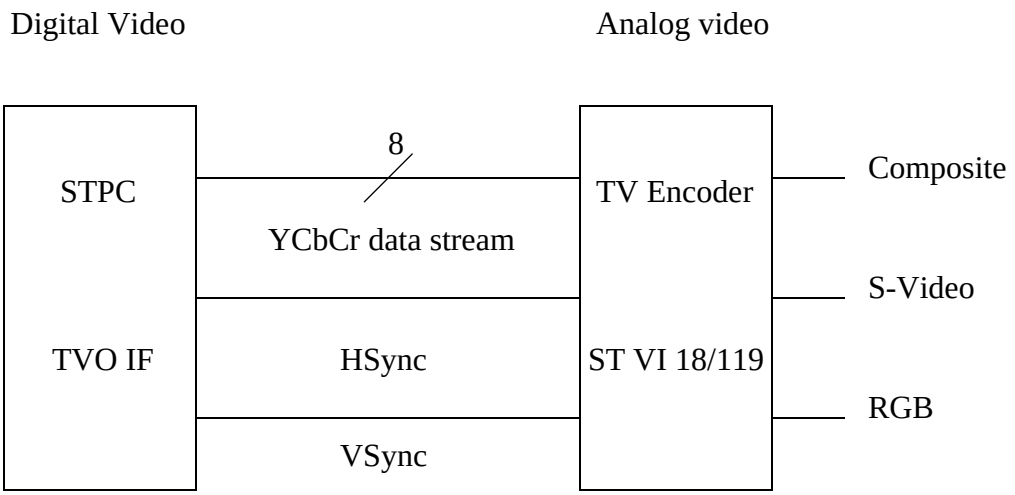
SR[0x02] = 0x0F (access to all the planes)
SR[0x02] = 0x0E (chain 4)
GR[0x02] = 0x00
GR[0x03] = 0x00
GR[0x04] = 0x00
GR[0x05] = 0x40
GR[0x06] bit [3,2] = 01 (A0000 to AFFFF)
eventually bit [3,2] = 00 (A0000 to BFFFF) but MDA will stop working.
GR[0x08] = 0xFF (bit mask)
CRTC[0x1C] bit 2 = 1 (sequential chain 4)

With these settings in place, the windowed access is enabled in CRTC extended register 0x1C bit 1. The two windows are 32K wide, the first is located at A000:0000, the second one is located at A000:8000. The offset in the frame buffer where the window is opened is defined by CRTC extended register 0x1D and respectively 0x1E for page 2. The value put into these registers defines the offset in a 32K Base.

Win1 FB offset = CR0x1D * 0x8000.
Win2 FB offset = CR0x1E * 0x8000.

6. TV Output

6.1. Overview



6.2. Synchro mode

With SIP100 there is also a possible synchronization mode, either CCIR656 (sync code embedded in the data stream) or CCIR 601 that use V and H sync signals.

CCIR656: The TVOut CCIR656 sync code in data stream must be enabled (set CR40 bit 6) since the H and V sync are in this mode not used those 2 pins of the STPC should be set as input to avoid any conflict with the digital encoder that might drive them (done by clearing CR40 bit 4 and 3).

CCIR601: The CCIRR656 should be turned off but is not required (clear CR40 bit 6). The STPC TV HSync and Vsync pins (might have different name VTV_bottom_top or odd_even and VTV_HSync) must be set to output (Done by setting CR40 bit 4 and 3).
Obviously these pins of the digital encoder must be linked to the STPC. Depending on board this might not be possible or will need specific jumper configuration. So check the hardware!

6.3. TV settings

Fixed parameter
When TV output is enabled many timings are fixed or must remain nearly at a given value to match the TV standard timings in terms of line and pixel per line. But the most important is the dot_clock that must be exactly 27MHz to give the exact HSync and VSync frequencies. The internal PLL is not accurate enough so it might result in an absence of colors on some TV Sets.

It depends on the TV set itself and on its tolerance of the 27MHz stability. So it is highly recommended to use external dot_clock with an 27MHz oscillator just to guarantee colors with all the TV set.

To make the flicker filter work in all the possible configurations the VRE must be set to a minimal value of 9, otherwise for 3 tap filter the image will become dark.

For PAL here is the parameter and its values:

Horizontal Total = 864
Vertical Total = 625 (624 is the nearest multiple of 8)
Horizontal display End: <720 (active resolution)
Vertical display End: <576 (active resolution)

For NSTC-M:

Horizontal Total = 858
Vertical Total = 525 (524 is the nearest multiple of 8)
Horizontal display End: <720 (active resolution)
Vertical display End: <485 (active resolution)

6.4. TV output specific registers

- CR40 Hold many TV output feature enable/disable bits and the flicker filter settings (see CR40 definition).

Typical value (assuming Flicker = 010b 1:2:1 3 tap filer)

	CCIR656	CCIR601
CR40	OxE2	OxFA

-CR41,CR42: Those 2 registers define the TV Active Video start (TV_AVS), which is the positioning in an pixel granularity of the VGA left corner of the TV screen. An Odd or Even value for this parameter might swapped the R And B colors (inverted in YCbCr data stream).
The purpose of these registers is to place the VGA on the left border of the TV.

This parameter should verify:

TV_AVS< HTOT-HDE-TV_HSE (remaining non active pixels in one TV line)

These parameters are accessible in Vgainfo, to avoid the color swap problem mentioned above, it can be changed by step of 2 pixel.
-CR43,CR44: These 2 registers define the width in pixels of the TV (interlaced) HSync pulse and is named TV Horizontal Sync End (TV_HSE). A typical value is 0x15;

6.5. Digital Encoder

In order to have a working TV output and improved quality, the STv0118/STv0119(*2) TV encoder must be initialized and correctly setup according to the board, chip capability and the desired operating mode.

The registers that require reprogramming are listed below. For further explanation, please refer to the STv0118 or STv0119 datasheets:

Register 0:

This register hold the synchronization mode and the TV standard.

Register 2:

This register is always set to 0x61.

Register 1 & Register 3:

Those two registers control trap filter. Selecting the right filter and enabling it to contribute to an i improvement of TV output quality:
:

	PAL	**NTSC**
STV-1	10xF4	0x54
STV-3	30xCO	0x80

Register 5:

Select which type of output is driven in the 3 additional analog line (select RGB or YC), it also allows us to invert the signal in case output is inverted at the amplifier stage (as is the case with all the ST evaluation boards).

	RGB	**S-Video**
STV-5	0xB1	0x31

(*2) The STv119 is STv118 + MACROVISION™

6.6. I2c Bus

The STv011x is programmed using an i2c bus, which can be implemented differently from board to board. The given Stvreg utility has been built for EVMINI and GLORIA boards, the i2c is assumed to be implemented using the STPC GPIO where the i2c_data is bit 2 and i2c_clock is bit 3. It is also assumed that the stv11x reset will be connected to the same GPIO bit 1.
In there is a different bit assignment within the GPIO, or because it is implemented on another GPIO the Stv utility and Setm program must be changed to take this into account.

For redefining the bits assignment changed the define in the beginning of Stv.c, those #define are in fact the mask and not the bit number:

```
#define SET_STV_RESET2
#define SET_STV_SDA4
#define SET_STV_SCL8
```

For changing the GPIO access method modify the Readi2cPort and Witei2cPort in Stv.c. As an example GPIO in use for i2c is custom, read io is at address 0x320 and write io at address 0x321.

```
#define I2C_READ_ADDR  0x320
#define I2C_WRITE_ADDR 0x321
void Writei2cPort(byte bData)

       IOWRITE8(I2C_WRITE_ADDR, bData);

byte Readi2cPort(void)

       return IOREAD8(I2C_READ_ADDR);
```

The stv11x reset may not be linked to any GPIO. In this case performing a soft reset is recommended. To enable the soft reset just comment the STV_HARD_RESET define in Stv.c. The software reset of the STv encoder is done using STv register 6 bit 7.

7..Generating a custom graphic mode

The easiest and fastest way to do this is to do it interactively, by playing with the various settings and directly seeing the results on the screen.

But which screen?

7.1. Working on VGA without VGA

Since we are going to modify and customize the vga display it is not possible to use it as the user interface. In doing so any operation done by the VGA bios may override previous user wanted changes,. Also when the resolution is changed in purpose bios will fail to correctly display anything on the screen. So an alternative display must be used. The monochrome adapter is perfect since it can be used at the same time as a VGA one. Under DOS to switch from VGA to mono just type "mode mono", to come back to the VGA "mode co80". Note that to setup a mode in the VGA using the bios it must be the active screen. It is also possible to use a terminal emulation, but the program must run on the target system.

7.2. How to start

The main idea is to build a custom mode interactively. This requires an STPC working system able to run the mode and the provided tools and an alternate display.

Typically it will be an EVMINI or another evaluation board with a BIOS, and Add On MDA or Serial terminal. These systems should be set up as the final platform (graphic clock, dram timing, TV out mode/type). This is not a requirement but it simplify everything.
The advantage of this system and method is to have all the TV support and VGA standard mode as a starting point, and being sure it will work on the final platform.

7.3. Starting from a existing mode

The VGA controller is almost complicated, so we should start with the nearest existing mode and adapted it step by step.
It’s highly recommend to use the same color depth, if it is not all the parameters for RAM-DAC&lookup-table,color format, line size,... etc, will have to be programmed separately one by one (there are 256*3 entry in the look up table).

Use as far as possible a VESA mode because must of the time they do not use standard VGA addressing mode with bits plane and other complex VGA stuff.
All the Standard mode support in VGA BIOS are listed below.

Mode	Type	PAGE	BPP	X	Y	Offset	C height
0000h	MDTYPE-CTEXT	8	4	40	25	80	16
0001h	MDTYPE-CTEXT	8	4	40	25	80	16
0002h	MDTYPE-CTEXT	8	4	80	25	160	16
0003h	MDTYPE-CTEXT	8	4	80	25	160	16
0004h	MDTYPE-CGA	1	2	320	200	80	8
0005h	MDTYPE-CGA	1	2	320	200	80	8
0006h	MDTYPE-CGA	1	1	640	200	80	8
0007h	MDTYPE-MTEXT	8	0	80	25	160	16
000Dh	MDTYPE-4BPP	8	4	320	200	40	8
000Eh	MDTYPE-4BPP	4	4	640	200	80	8
000Fh	MDTYPE-MGRAF	2	1	640	350	80	14
0010h	MDTYPE-4BPP	2	4	640	350	80	14
0011h	MDTYPE-1BPP	1	1	640	480	80	16
0012h	MDTYPE-4BPP	1	4	640	480	80	16
0013h	MDTYPE-8BPP	1	8	320	200	320	8
010Ah	MDTYPE-CTEXT	2	4	132	43	264	8
0109h	MDTYPE-CTEXT	2	4	132	25	264	16
0102h	MDTYPE-4BPP	1	4	800	600	100	16
0106h	MDTYPE-4BPP	1	4	1280	1024	160	16
0100h	MDTYPE-8BPP	1	8	640	400	640	16
0103h	MDTYPE-8BPP	1	8	800	600	800	16
0104h	MDTYPE-4BPP	1	4	1024	768	128	16
0105h	MDTYPE-8BPP	1	8	1024	768	1024	16
0101h	MDTYPE-8BPP	1	8	640	480	640	16
010Ch	MDTYPE-CTEXT	2	4	132	60	264	8
010Bh	MDTYPE-CTEXT	2	4	132	50	264	8
0108h	MDTYPE-CTEXT	6	4	80	60	160	8
0112h	MDTYPE-24BPP	1	24	640	480	2048	16
0107h	MDTYPE-8BPP	1	8	1280	1024	1280	16
0111h	MDTYPE-16BPP	1	16	640	480	1280	16
0114h	MDTYPE-16BPP	1	16	800	600	1600	16
0115h	MDTYPE-24BPP	1	24	800	600	3272	16
0117h	MDTYPE-16BPP	1	16	1024	768	2048	16
010Eh	MDTYPE-16BPP	1	16	320	200	640	8
010Fh	MDTYPE-24BPP	1	24	320	200	1024	8

Examples:

1)Want to build a custom PAL 720x540x8 TV mode
So lets start with VESA mode 101 640x480x8 (TV out set to PALDBGHI in BIOS setup).

2)Want to build a real 320x200x16 mode for TFT display.
Start with VESA mode 0x10E 320x200x16 (400 scan lines).

7.4. Changing the resolution

The active resolutions (active pixel by line or line per frame) are defined by HDE and VDE (V,H display end see clock mode).

The fastest method is when those values are far away enough from the corresponding retrace start, blanking start and mainly the Total parameters, it is then possible to slightly increase the resolution just by increasing the Display End, to make the new pixel visible the retrace start and blank start must be greater than the new display, so they should be incremented also.

For example if TV is enable the HTOT is around 860, so just by increasing HDE it is possible to increase the horizontal resolution, HRS and HBE must also be increase by the same amount to make the new pixel visible.

Note: Each time the resolution changed the memory mapping also changed so the screen must be repaint according the new parameters (use Inis).

8. Examples

Let’s start with mode 10E, 320x200x16

Horizontal resolution special care:

When using count by 2 or count by 4 mode the HDE must always be a multiple of 2 or 4 to ensure a integer number of pixel by line.

When H resolution changes the line Offset (size) must be changed according to the new number of pixel per line, when this is not done any pixels which are added become duplicates of the first one.

When the resolution is decreased the offset can remain the same (more memory than really used per line), adjusting it just to save graphic memory.

Whenever the offset changes so does the memory organization and it might be reinitialized to have the correct screen.

Typically, the first step is to modify H resolution (HDE), then the offset is adjusted, finally the screen is repainted using Inis.

Vertical resolution special care:

Before increasing the V resolution ensure that the Frame buffer is large enough to hold all the pixels.

Maintaining refresh rate:

Whenever the HTOT/VTOT changes it changes the H/V Sync period and H/V refresh rate, so to keep them 'unchanged' the dot_clock should move on the same ratio. This means:

$$New\_Clock = Old\_Clock \bullet \frac{New\_H/VTOT}{Old\_H/VTOT}$$

8.1. Standard mode Special refresh

For some special applications it might be useful to change the refresh rate of an existing mode, either to match a specific display rate, or just to reduce the memory band width used by the vga (more for CPU). It is really easy, all that is required is to reprogram the dot_clock to achieve the new refresh using the formula below:

$$New\_Clock = Old\_Clock \bullet \frac{New\_Refresh}{Old\_Refresh}$$

Warning: The correct synthesizer has to be reprogrammed. Also when using the vga bios, synthesizer 0 and 1 should not be changed or it will change the refresh for all standard modes and some VESA ones. It is better to select synthesizer 2 or 3 (see Miscellaneous ouput register) and then reprogram it.

8.2. Special mode with given refresh

It is more complicated and requires adapted timings and perhaps special clocking mode to get wanted V and H refresh.

The goal is in fact to compute dot_clock to match Wanted HSync VSync given Hres Vres. For example if we have Hsync_Freq, Vsync_Freq.

The Vertical total number can be calculated with the following formula:

$$VTOTAL = \frac{Hsync\_Freq}{VSync\_Freq}$$

At this point we have one limitation on the Vertical resolution that cannot be more than around VTotal -12.

To compute dot clock multiply Hsync_Freq by HTOT, (HTOT is at least around HRes + 30):

Dot_clock#HSync_freq*(Hres + 30).

As an example screen sync rates are 32KHz H and 50Hhz V, 480x480x8 mode.
Vtotal = 32000/50 = 640
Htotal = 480 + 30 = 510.
Dot_clock = 32000 * 510 = 16.3MHz.

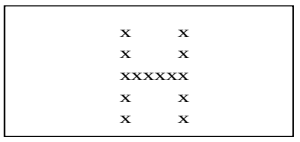
So the graphic mode resolution may be up to 480x600, but it can be changed, if required, to a resolution of around 320x200. The vertical line will have to be at least double (320x400) or triple (320x600) to match the VSync spec.

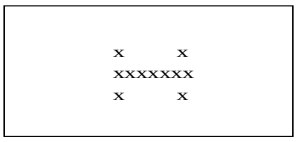
To complete the setup of the VGA, we need to program the following parameters:

CCH = 1	HRS = 488
HTOT = 510	HRE = 24
HDE = 480	VTOT = 640
HBS = 480	VDE = 480
HBE = 8	VRS = 500

8.3. Double Line to Single Line mode

Many 200 line modes are in fact displayed as 400 line modes because standard VGA or SVGA display cannot synchronize with only 200 lines (16KHz H refresh).
These modes use the scan double feature to duplicate each line, if the scan double (bit7 in CR9) is disabled the same display can be seen on screen but only using half of the screen height, the missing pixel are black. Note that just by putting a CCH = 2 the image will appear normal again it as the same effect.





To get back to a full height display by using only 200 lines, we have to modify all the vertical timings to about half their current values.

In the example below we have started from mode 0x10E 320x200x16 we have;

	400 lines	**200 lines**
VTOT	447	247
VDE	400	200
VRS	412	212
VRE	11	11
VBS	406	206
VBE	57	117

Once these value have been set the image disappears on the VGA monitor, but it is normal, we have divided the Vsync freq by 2 and the dot_clock now has to be divided by 2 also. By reading miscellaneous out register we have found that the synthesizer in use is 0, so we use Dclk utility in synthesizer 0

Dclk 0
M 14 N 197 P 3
PLL 0 (76,95)=25.184Mhz
Dclk 0 12.5

still nothing, it is normal VGA or SVGA display cannot synchronize with only 200. On some hi performance VGA monitor it will be possible to synchronize with dclk = 20MHz
(h refresh = 27KHz V refresh = 110Hz)

8.4. Mollifying TV mode resolution

Vertical resolution

1) To use TV output restart the computer enter the system bios setup, enable the TV by setting PAL DBGHI for example, also make sure the frame buffer size is large enough to support the custom mode we will build, let's put 4 Meg for the trial.

2) Setup the graphic mode, let use VESA mode 0x101 640x480x8.
-> 'Vmode 101'

3) Now switch to mono display.
-> 'mode mono'

4) Setup the vga overscan color to something other than black so that we can see the blanking.
->'Over 30'

5) Modify the V resolution let us try to build a 525 lines mode.
To modify all the parameters use Vgainfo.exe.
-Increase VDE up to 525 (524 in register).

6) Adapt vertical timings.
Not all the added lines can be seen? This is normal, the news lines were never initialized in memory, also other timing must be changed.
->'Inis'. to init the screen by taking care of the new V resolution.
Are all the lines still not visible? Modify other V timings.
-Set VBE to 0, so blank does not bother
-Set VBS to 527 so we have 2 lines blank as a marker.

7) Shift the image to the top
Now we start increasing VRS, as VRS increases the image is shifted to the top, finally the full image is visible when VRS = 564.

Horizontal resolution

We will continue or custom mode and set the H resolution to the max for PAL so it's 720.

8) Program the resolution
-Set VDE to 720.
Something strange happens, some new pixels can be seen but not all. Also it looks like the last pixels are replications of the 1st in the lines.
This is normal. The line size has changed and it must be setup,
Set the offset, must put at least the line size
-Set offset to 0x5A (720/8)
->'Inis' To setup the screen according to the new memory line size/resolution.

9) Adapt other Horizontal timings.
-Set HBE = 0
-Set HBS = 728 (more than VDE).

10) Now shift the image to the left
By increasing HRS we can see the image shifting left, at some point we cannot obtain something better by changing HRS. Try to decrease TV_AVS, then try HRS again and so on until satisfied. By changing TV_AVS color may swap by adding one the good color should come back again.
Finally we have: HRS = 728.

Save info:
To save these settings let us run Genmode without argument.
-> Genmode
The file mode.h has been generated, recompile Setm, and use it now to setup these customs 720x525x8 mode.

Important Trick:

-VRS can be used to position vertically the active screen.
-HRS/TV_AVS can be used to position horizontally the active screen.
-When wrong (swapped) colors are displayed increase or decrease TV_AVS by 1.

9. Using the tools

9.1. Genmode.exe (generate mode report)

Genmode capture VGA settings and save them to a ’.h’ and ’.inc’ file. Those to file can then be use to compile the Setm.c (or Setm.asm) program to setup the vga back.
Genmode can use the vga bios to setup a standard mode and capture it.

Genmode /mMode_number

this is the same as doing:

Vmode Mode_number
Genmode

Genmode have command line options to control the issue of clock and dram.
- The ’k’ option will add in .h file a flag the Setm.c will not setup the clock.

-The ’d’ option
To skeep CRTC[0x3C] in .h file so gdram timings will not be programmed

-edo60,edo70,fpm60, fpm70
To force gdram type and speed.

9.2. Vgainfo.exe

It’s an ugly user text interface, but it allow you to quickly set parameters and timings. To start it use ’Vgainfo vga.reg’, the vga.reg is a text file with the parameters description.

The screen will appear as following:

```
Pixel format 0x00000001 1
Character Cell Height (add 1)  0x0000000F 15 + 1 = 16
Horizontal Total (-5)  0x0000005F 95 + 5=100 * 9 = 900
Horizontal Display End (-1)  0x0000004F 79 + 1 = 80 * 9 = 720
Horizontal Blanking Start  0x00000050 80 * 9 = 720
Horizontal Blanking End (pulse width)  0x00000022 34 * 9 = 306
Horizontal Retrace Start  0x00000055 85 * 9=765
Horizontal Retrace End (pulse width)  0x00000001 1 * 9 = 9
Vertical Total 0x000005BF 1471
Vertical Display End  0x0000058F 1423 + 1 = 1424
Vertical Blanking start 0x00000596 1430
Vertical Blanking End (pulse Width) 0x00000039 57
Vertical Retrace Start 0x0000059C 1436
Vertical Retrace End (pule width) 0x0000000E 14
Line Compare 0x000003FF 1023
Offset (byte Line size) 0x00000128 296
Start address (1st pix addr) 0x00010000 65536
TV Horizontal Active Video start 0x00000204 516
TV Horizontal Sync End 0x00000302 770
```

There is one parameter per line, the one selected as the '*'.

The first displayed data is the one defined by register in hexa, then there is the same information in decimal, an additional offset may be added, finally the value may be multiplied by the character width and displayed as a decimal.

Command:

To change the selection use the **Direction arrow (up down)**.
To increase by 1 use '+'.
To decrease by 1 use '-'.
To enter the new value with the keyboard, type **enter**
The data format is decimal (by default), hexa start by 0x, octal start by 0, binary start by 0b.

Data entered directly with the keyboard must not consider character width multiplier or additional offset.

9.3. Setm.c

This is the source file that will set a a custom mode, it used the registers dump done by Genmode (mode.h). Setm.c that does not use any OS specific nor standard library call, so it should run with any OS and C ANSI compiler.
All the IOs are done through IOWrite8/16 and IORead8 functions implemented in misc.c. These IO functions should be replaced by the OS/Compiler/Platform specific.
Warning Setm does not perform STv011x programming for TV output mode.

Note:

The code has been tested so it works! It is currently used in Win-CE and other embedded applications to setup graphic mode.

9.4. Dclk 1.3 Tool

The Dclk utility program can be used to program or retrieve the value in the onchip clock synthesizers. It is really simple and only uses command line options:

To get the value in the generator 0: Dclk 0
To get the value in the generator 3: Dclk 3
To put 25.1MHz in generator 2: Dclk 2 25.1
To put 28.215MHz in generator 0: Dclk 0 28.215
To get the value in currently use generator Dclk *x*
To set the value in currently use generator Dclx *x*